An Efficient Unique State Coding Algorithm for Signal Transition Graphs *

Enric Pastor and Jordi Cortadella
Department of Computer Architecture
Universitat Politècnica de Catalunya
08071 Barcelona, Spain

Abstract

Current algorithms to force the Complete State Coding (CSC) property for Signal Transition Graphs work on the State Graph and, therefore require exponential time and space. Polynomial algorithms have been only proposed for Marked Graphs. In this paper, a P-time algorithm for Unique State Coding (USC) is presented. Although more restrictive than CSC, it is shown that the USC property can be efficiently guaranteed for large STGs. Several experiments evidence that the obtained results are even better than those generated by exponential-time techniques.

1 Introduction

Signal Transition Graphs (STGs) are a subclass of interpreted Petri nets originally presented in [1] for the specification of asynchronous circuits. Since then, different synthesis techniques have been proposed in order to automatically produce hazard-free implementations [9, 7].

One of the necessary conditions for an STG to be implementable is the Complete State Coding (CSC) property, which guarantees that different states can be disambiguated by only considering the information maintained in their coding. The Unique State Coding (USC) property, similar to CSC but still more restrictive, has been also used for the same purpose.

Table 1 summarizes the most significant efforts in the area of CSC for Petri nets. A common characteristic of these techniques is that the State Graph (SG) derived from the net is the main data structure used to analyze the CSC property (excepting [12] for Marked Graphs (MGs)). Since the size of the SG can be exponential depending on the number of signals of the circuit, algorithms using the SG show an exponential time complexity for their worst case.

	MG	FC net	Petri net	*time*
VANBEK [12]	×			$O(n^4)$
KISH [6]	×			exp.
LAVAG [8]	×	×		exp.
VANBEK [13]	×	×	×	exp.
This paper	×	×		$O(n^4)$

Table 1: Previous techniques for CSC of Petri nets

In this paper, we present a polynomial-time algorithm to satisfy the USC property of STGs. The proposed technique does not require the SG to detect coding conflicts. It is based on the analysis of a subset of State Machines (SMs) that *cover* the STG and the insertion of state signals to disambiguate potential USC conflicts. The reason why we use USC instead of CSC is that CSC requires a more exhaustive analysis of the STG that would make the algorithm exponential. In spite of using a more restrictive property, we have found that our algorithm can yield better results that those obtained by existing exponential-time techniques.

The main motivation that led us to explore P-time USC algorithms is the need for synthesizing large and complex STGs, automatically generated by High-Level synthesis tools [2, 3], that would require a substantial amount of CPU time with exponential-time algorithms.

The paper is organized as follows. Section 2 describes how potential USC conflicts can be detected on the STG. Section 3 describes the state signal insertion procedure used by our technique. Section 4 analyses the complexity of the proposed algorithms. Section 5 presents some experimental results. Finally, conclusions and future work are presented in section 6.

2 USC Property Analysis

To be consistent with the nomenclature used by other authors, we have imported most of the preliminary definitions and results from [1, 5, 8, 10, 12].

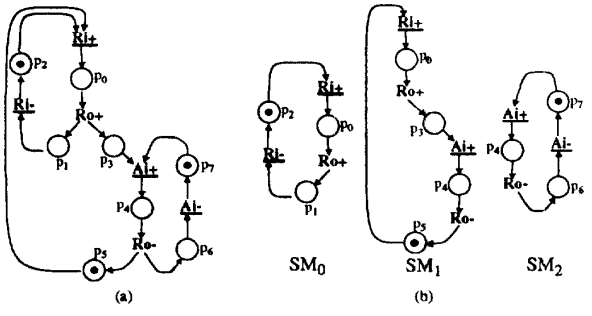

Figure 1: (a) STG Specification of a PLA Interface Circuit, (b) SMCs of the PLA Interface Circuit

Figure 1(a) depicts an example of STG that will be used along this paper. The example [8] specifies the behavior of a PLA Interface Circuit. Figure 1(b) shows all the SM-Components (SMCs), while the State Graph with its USC conflicts is shown in figure 2.

*Work funded by CYCIT TIC 91-1036, ACiD-WG (Esprit 7225) and Dept. d'Ensenyament de la Generalitat de Catalunya

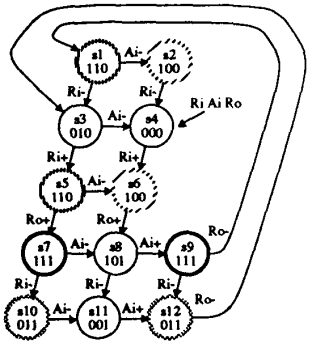

Figure 2: State Graph of the PLA Interface Circuit

2.1 USC Property at STG-level

The first formal definition of the USC property at STG-level for *Marked Graphs* was given in [1]. In this section we present an extension of Chu's definition for FC nets[1].

Definition 1 *A non-empty set of transitions is said to be feasible if there exists a marking from which these transitions can be fired without firing a transition not belonging to the set.*

Definition 2 *A set of transitions is a complementary set iff it contains the same number of rising and falling transitions for each signal.*

Definition 3 *A set of SM-Components (MG-Components)$\{S_i\}$, is a cover of an STG S if all places and transitions of S are covered by some S_i. A cover $\{S_i\}$ is irredundant if no subset of $\{S_i\}$ is a cover.*

Theorem 4 [11] *A live and safe STG S has two markings m_i and m_j with the same coding iff exists a feasible complementary set, and the set does not cover any MG-Component (MGC) in S.*

2.2 USC Checking Procedure

Definition 5 *For each place p_i of a live and safe FC (LSFC) net, we define Π_i as the set of markings which have a token in p_i, i.e. $\Pi_i = \{m_k \in [m_o\rangle \mid m_k(p_i) = 1\}$.*

Since every marking has a one-to-one correspondence with a state of the State Graph, we can similarly make the following definition for binary vectors.

Definition 6 *For each place p_i of a LSFC net, we define $\mathcal{V}_i$ as the set of binary vectors of the markings which have a token in p_i, i.e. $\mathcal{V}_i = \{v_k \mid m_k \in \Pi_i\}$.*

The following theorem presents *necessary and sufficient conditions* for USC.

Theorem 7 [11] *Let S be a live and safe STG and Σ a set of SMCs that irredundantly covers S. Then S has the USC property iff $\forall$ SM $\in \Sigma$ and $\forall p_i, p_j$ covered by SM, $i \neq j \Rightarrow \mathcal{V}_i \cap \mathcal{V}_j = \emptyset$.*

Theorem 7 states that selecting a set of SMC that irredundantly covers all the places of S, and intersecting the $\mathcal{V}_i$ sets in each SMC, it is possible to find all the USC conflicts in the STG. The number of SMC in the Σ set is limited by the number of places in S.

[1]The original extension proposed by Chu is incorrect, as shown with a counterexample in [11]

p_i	Π_i	$\mathcal{V}_i$ $(R_i A_i R_o)$	$\mathcal{C}_i$
p_0	{m5, m6}	{110, 100}	1 - 0
p_1	{m1, m2, m7, m8, m9}	{110, 100, 111, 101}	1 - -
p_2	{m3, m4, m10, m11, m12}	{010, 000, 011, 001}	0 - -
p_3	{m7, m8, m10, m11}	{111, 101, 011, 001}	- - 1
p_4	{m9, m12}	{111, 011}	- 1 1
p_5	{m1, m2, m3, m4}	{110, 100, 010, 000}	- - 0
p_6	{m1, m3, m5, m7, m10}	{110, 010, 111, 011}	- 1 -
p_7	{m2, m4, m6, m8, m11}	{100, 000, 101, 001}	- 0 -

Table 2: Π_i, $\mathcal{V}_i$, and $\mathcal{C}_i$ for the PLA Interface Circuit

In the example, $\mathcal{V}_3 \cap \mathcal{V}_4 \neq \emptyset$ for SM_1 evidences the existence of a USC conflict involving places p_3 and p_4.

2.3 P-time USC Checking Algorithm

Finding $\mathcal{V}_i$ for each place p_i requires the generation of the complete reachability set $[m_o\rangle$, which can have exponential size. Instead of calculating $\mathcal{V}_i$ for each place p_i, we will find a cube $\mathcal{C}_i$ that covers all the binary vectors in $\mathcal{V}_i$. Thus, if $\mathcal{C}_i$ and $\mathcal{C}_j$ are two cubes that cover $\mathcal{V}_i$ and $\mathcal{V}_j$ respectively, then $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset \Rightarrow \mathcal{V}_i \cap \mathcal{V}_j = \emptyset$. Therefore, the USC property can also be checked, according to theorem 7, by comparing cubes. However, note that USC checking by cube comparison is more restrictive, i.e., more potential USC conflicts can be detected even if they do not exist. In spite of this, the experiments showed that this restriction hardly influences on the quality of the results.

Next, we describe how the cubes can be calculated in polynomial time.

2.3.1 Calculation of Covering Cubes

Definition 8 *Let S be a LSFC net. A pair $(t \in \mathcal{T}, p \in \mathcal{P})$ is called to be* concurrent *$((t,p) \in$ co$(S))$ if $\exists m_n \in [m_o\rangle$ such that t is enabled, $m_n(p) = 1$ and t is not an output transition of p. Otherwise they are in* conflict *$((t,p) \in$ cf$(S))$.*

Given an SMC SM of S and a place p_i covered by SM, let us call $\mathcal{C}_i$ the cube that covers all the binary vectors in $\mathcal{V}_i$, and $\mathcal{C}_i^j$ the $j-th$ component of $\mathcal{C}_i$, corresponding to the signal t_j of S. If all the transitions of the signal t_j are in *conflict* with the place p_i, then all the binary vectors $v \in \mathcal{V}_i$ will be either $v^j = 1$ or $v^j = 0$ and consequently $\mathcal{C}_i^j = v^j$. To determine the value of $\mathcal{C}_i^j$, we must find the first transition t_j^* enabled after firing some transitions from a marking m_p where place p_i was marked. If t_j^* is a rising transition then $\mathcal{C}_i^j = 0$, otherwise $\mathcal{C}_i^j = 1$. If some transition of signal t_j is *concurrent* with p_i, then $\mathcal{V}_i$ will have some binary vectors with $v^j = 1$ and the rest with $v^j = 0$, and therefore $\mathcal{C}_i^j = -$.

Table 2 shows the set of markings, binary vectors and cubes for each of the places of the example.

3 Insertion of State Signals

3.1 USC Conflict Graph

Definition 9 *For each SMC SM of an STG S we define the* Conflict Graph *CG(SM) as the pair $< \mathcal{P}, \mathcal{U} >$, where $\mathcal{P}$ is the set of places of SM and $\mathcal{U} \subset (\mathcal{P} \times \mathcal{P})$ is the USC conflict relation, defined as follows:*

$$(p_i, p_j) \in \mathcal{U} \Leftrightarrow \mathcal{C}_i \cap \mathcal{C}_j \neq \emptyset.$$

Given a set of SMCs, Σ, that covers S, the *Conflict Graphs* derived from Σ define all pairs of places with potential USC conflicts. The technique we propose is based on inserting state signal transitions for each SMC with conflicts. However, the information given by all the CGs is redundant, as we will show with the following theorem.

Theorem 10 [11] *Let S be a live and safe STG and Σ a set of SMCs that irredundantly covers S. Then S has the USC property if $\forall p_i \in S \; \exists \; SM_k \in \Sigma$ that covers p_i such that $\forall p_j \in SM_k, i \neq j, \; C_i \cap C_j = \emptyset$.*

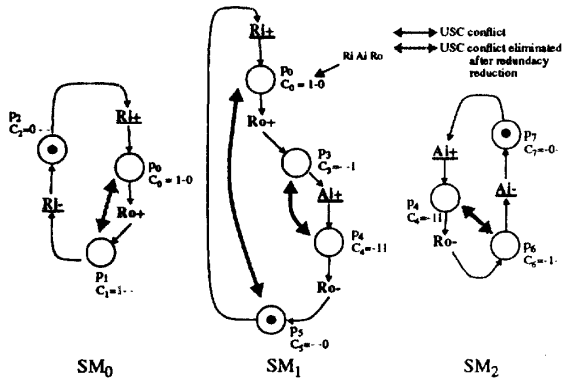

Figure 3: Potential USC conflicts for each SMC.

From the previous theorem we can deduce that if for a place p_i we can find an SMC without any edge (p_i, p_j) in its CG, then p_i does not generate any USC conflict. Therefore, all the edges (p_i, p_k) in the rest of CGs can be eliminated.

In [11], an $O(n^3)$ heuristic is presented to reduce the redundancy of the CGs iteratively including those SMC that cover the maximum number of conflicts not covered yet by other SMCs. Figure 3 depicts the potential conflicts for each SMC of the example, and the edges of the Conflict Graph eliminated after redundancy reduction.

3.2 State Signal Insertion Procedure

After redundancy reduction, signal transitions are inserted for those SMCs that still have potential USC conflicts. Each inserted state signal partitions the SMC into two sets of places.

Definition 11 *A place p_i of an SMC is in the* On-set *(*Off-set*) of a signal t_j, if p_i is located between a rising (falling) and a falling (rising) transition of t_j.*

A potential USC conflict in an SMC can be disambiguated by inserting an state signal in such a way that each of the involved places is located in a different set. Moreover, the same signal can be used to disambiguate multiple conflicts at the same time.

3.2.1 SMC Bipartitioning

To insert an state signal t in an SMC, the set of places must be partitioned into the On-set and the Off-set of t.

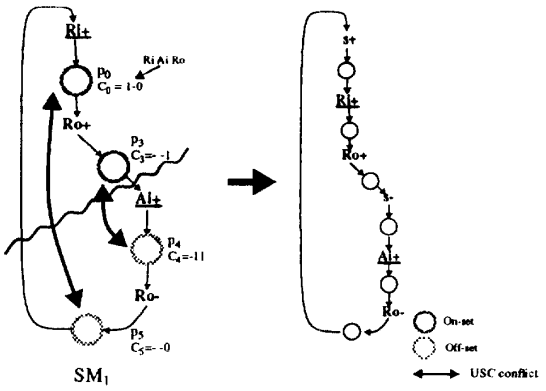

Figure 4: SMC partitioning and state signal insertion.

Then, a transition $t+$ $(t-)$ must be inserted for each pair of consecutive places (p_i, p_j) such that $p_i \in$ Off-set(t) (On-set(t)) and $p_j \in$ On-set(t) (Off-set(t)) (note that more than one pair of $t+$ and $t-$ transitions may be required).

In our approach we use a Fiduccia&Mattheyses-based algorithm [4], with a cost function that aims at maximizing the number of conflicts disambiguated with one state signal and, minimizing the number of transitions of the state signal. State signals are inserted until all the potential USC conflicts are eliminated.

Figure 4 shows how an SMC component is partitioned and an state signal inserted to eliminated USC conflicts in the example. The obtained solution (figure 5) is the same than that presented in [8].

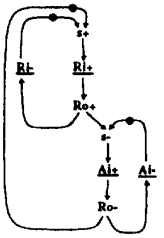

Figure 5: STG with the USC property

4 Complexity Analysis

Table 3 summarizes the complexity analysis for each step of the procedure for solving USC conflicts. For simplicity, we have considered $n = \max(|\mathcal{T}|, |\mathcal{P}|)$. The time complexity is dominated by the calculation of the cubes, which is $O(n^4)$.

5 Experimental Results

The examples used for our experiments have been obtained from [8]. Moreover, we have also included some STGs automatically generated by high-level synthesis tools [2, 3] (*gcd_alu*, *gcd_reg_a*, and *gcd_reg_b*).

Table 4 presents the results obtained by [8] and our approach. The number of signals, transitions, and SG states

	inital				final [8]			final			
STG	sig	tr	states	SM	sig	tr	lit	sig	tr	lit	time(sec)
alloc-outbound	7	18	17	1	9	22	19	9	22	26	0.6
atod	6	12	20	3	7	14	14	7	14	13	0.2
nak-pa	9	18	56	4	10	22	30	10	20	24	0.7
ram-read-sbuf	10	20	36	4	11	22	20	11	22	24	1.0
sbuf-ram-write	10	20	58	3	12	24	30	12	24	23	1.4
sbuf-read-ctl	6	12	14	2	7	14	13	7	14	15	0.3
sendr-done	3	6	7	2	4	8	5	4	8	8	0.1
vbe4a	6	12	76	3	8	16	22	8	16	31	0.3
vbe6a	8	16	128	3	10	20	30	10	20	38	0.7
master-read	13	26	8932	8	16	36	77	15	30	46	2.5
gcd_alu	8	16	228	4	-	-	-	11	22	35	0.7
gcd_reg_a	20	58	1274	5	-	-	-	25	75	84	29.9
gcd_reg_b	18	54	1146	5	-	-	-	23	69	97	21.5

Table 4: Experimental Results

Step	time-complexity
cf(S) calculation	$O(n^3)$
C_i calculation	$O(n^4)$
SMC calculation [5]	$O(n^2)$
CG calculation	$O(n^2)$
State signal insertion	$O(n^3)$

Table 3: Complexity analysis for the USC procedure.

of the original STGs are presented in the columns labeled "initial". SM denotes the number of SMCs used by our procedure to cover the STG.

For all the benchmarks (except for *master-read*) the number of inserted signals is the same for both techniques. For *nak-pa* our procedure inserts fewer transitions. After forcing the USC property, we used SIS [7] to obtain the number of factored form literals. We show that good-quality results can be obtained with short CPU times. This results made evident the promising suitability of our approach for large STGs.

Finally, the last two rows present results on the largest STGs we have encountered so far. They represent a good sample of the type of circuits that can be automatically generated by synthesis tools. Both required five internal signals that were inserted in less that 30 secs.

6 Conclusions and Future Work

We have presented a P-time algorithm to solve the Unique State Coding problem for STGs. Unlike other techniques used for the same problem, only the STG representation is required for the algorithm. State disambiguation is performed by the insertion of state signals.

The results obtained from several experiments show that our approach, with short CPU times, improves the quality of the solutions obtained by other existing techniques.

Several open issues are left for the future. In the area of P-time algorithms, we are exploring more precise algorithms to detect conflicts, reduce the complexity of the synthesized circuit, and extend our approach to any class of Petri nets, especially for those synthesis systems that generate large and complex STGs automatically.

References

[1] Tam-Anh Chu. *Synthesis of Self-timed VLSI Circuits from Graph-theoretic Specifications.* Ph.d. thesis, MIT, June 1987.

[2] Jordi Cortadella and Rosa M. Badia. An asynchronous architecture model for behavioral synthesis. In *Proc. of the EDAC*, pages 307–311, March 1992.

[3] Jordi Cortadella, Rosa M. Badia, Enric Pastor, and Abelardo Pardo. Achilles: A high-level synthesis system for asynchronous circuits. In *Sixth International Workshop on High-Level Synthesis*, pages 307–311, November 1992.

[4] C.M. Fiduccia and R.M. Mattheyses. A linear time heuristic for improving network partitions. In *Proc. of the 19th DAC*, 1982.

[5] M. Hack. *Analysis of production schemata by Petri nets.* M.s. thesis, MIT, February 1972.

[6] M.A. Kishinevsky, A.Y. Kondratyev, and A.R. Taubin. Formal method for self-timed design. In *Proc. of the EDAC*, pages 197–201, February 1991.

[7] L. Lavagno, K. Keutzer, and A. Sangiovanni-Vincentelli. Algorithms for synthesis of hazard-free asynchronous circuits. In *Proc. of the 28th. DAC*, pages 302–308, June 1991.

[8] L. Lavagno, Cho W. Moon, Robert K. Brayton, and A. Sangiovanni-Vincentelli. Solving the state assignment problem for signal transition graphs. In *Proc. of the 29th. DAC*, June 1992.

[9] Teresa H.-Y. Meng, Robert W. Brodersen, and David G. Messerschmitt. Automatic synthesis of asynchronous circuits from high-level specifications. *IEEE Trans. on CAD*, Vol. 8(11):1185–1205, November 1989.

[10] Tadao Murata. Petri Nets: Properties, analysis and applications. *Proceedings of the IEEE*, Vol. 77(4):541–574, April 1989.

[11] Enric Pastor and Jordi Cortadella. P-time unique state coding algorithms for signal transition graphs. Technical Report RR-93/13, UPC/DAC, May 1993.

[12] P. Vanbekbergen. Optimized synthesis of asynchronous control circuits from graph-theoretic specification. In *Proc. of the ICCAD*, pages 184–187, November 1990.

[13] P. Vanbekbergen, B. Lin, G. Goossens, and H. De Man. A generalized state assignment theory for transformations on signal transition graphs. In *Proc. of the ICCAD*, pages 184–187, November 1992.